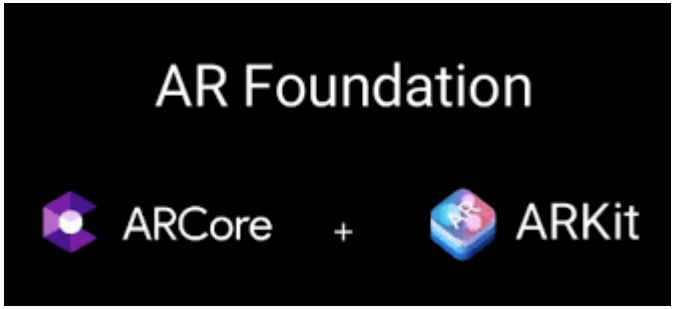
Unity3D and Augmented Reality

Main Menu

- Unity3D basics and editor layout
- Augmented Reality with AR Foundation

[1] - Unity logo



[2] – AR Foundation Logo

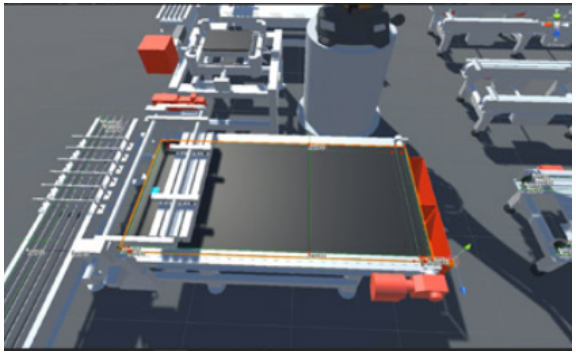
What is Unity3D?

- Gaming dreams into reality.
- Export to many platforms.
- C, C++ (Runtime), C# (Unity API)
- Drag and drop
- Big community

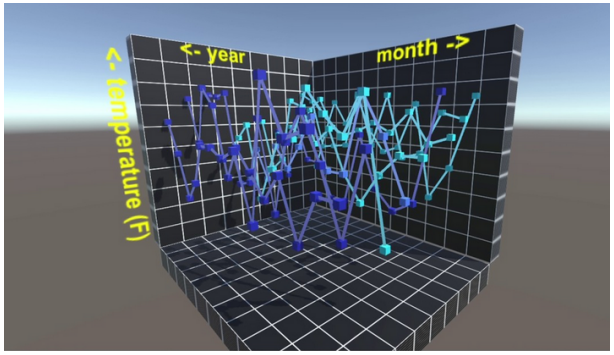
[3] - Unity-chan

What can you do?

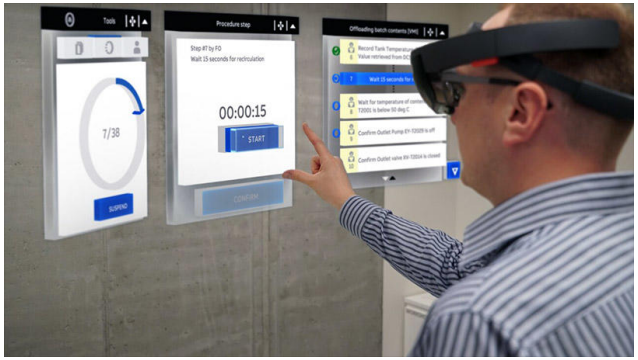
[4] Game



[5] Simulation



[6] Data visualize



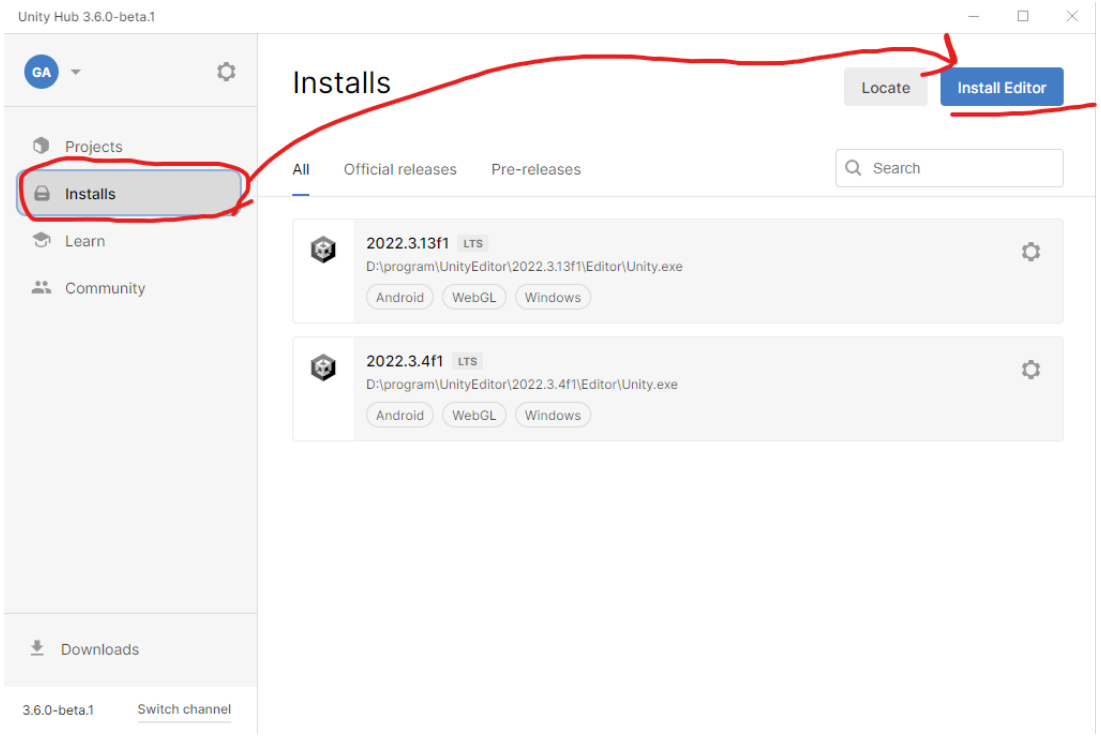
[7] AR/VR

[8] Film and Animation

[9] Education

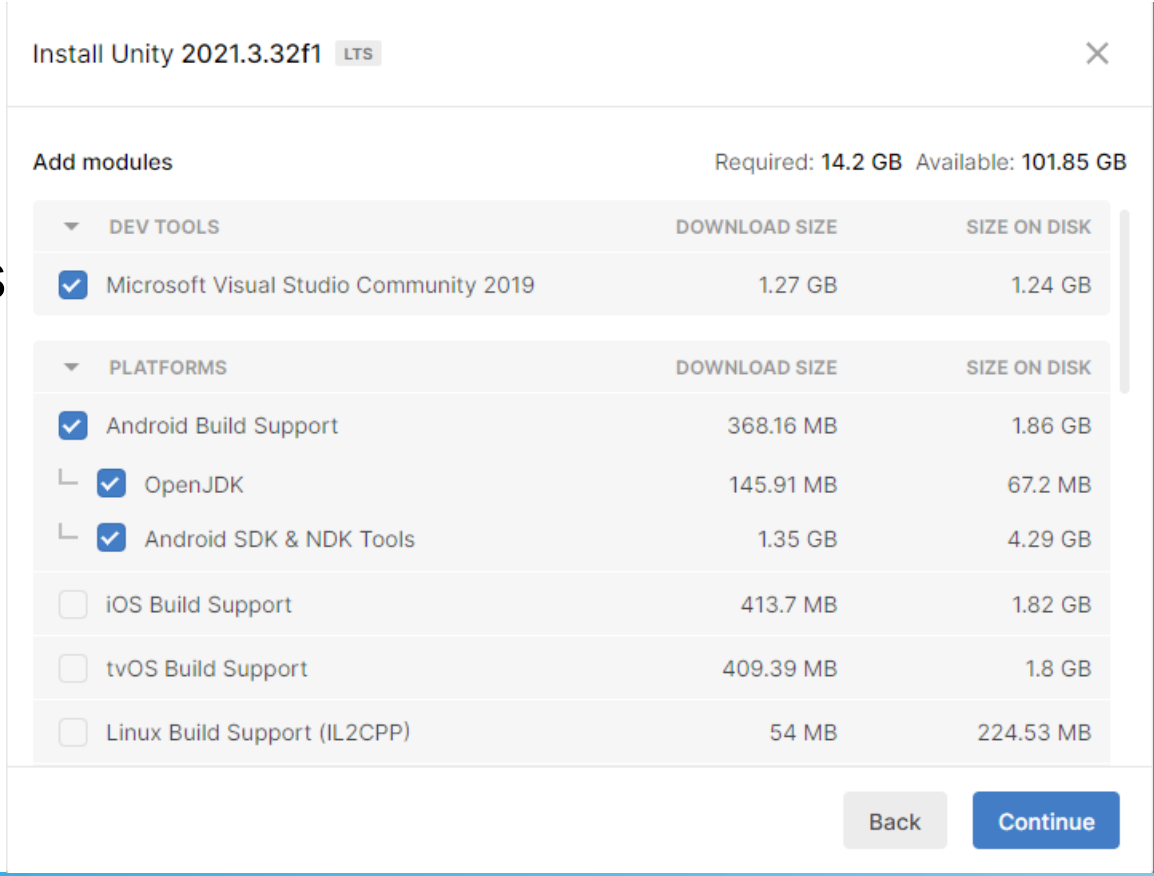
Download

- https://unity.com/download
- Download **unity hub**
- Sign up and activate a license
- Installs – Install Editor



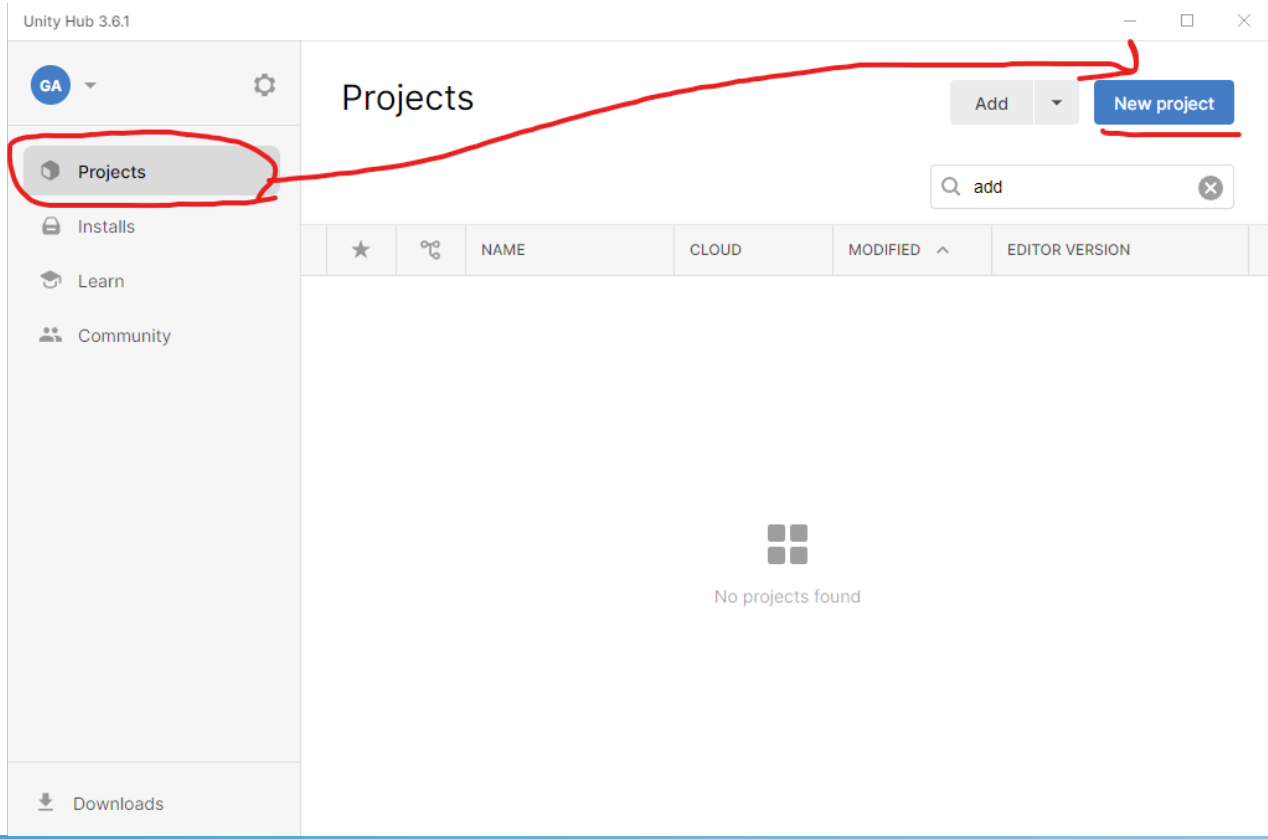
Install

- Select dev tool
- Select your target platforms
- Accept the required licenses
- And wait…



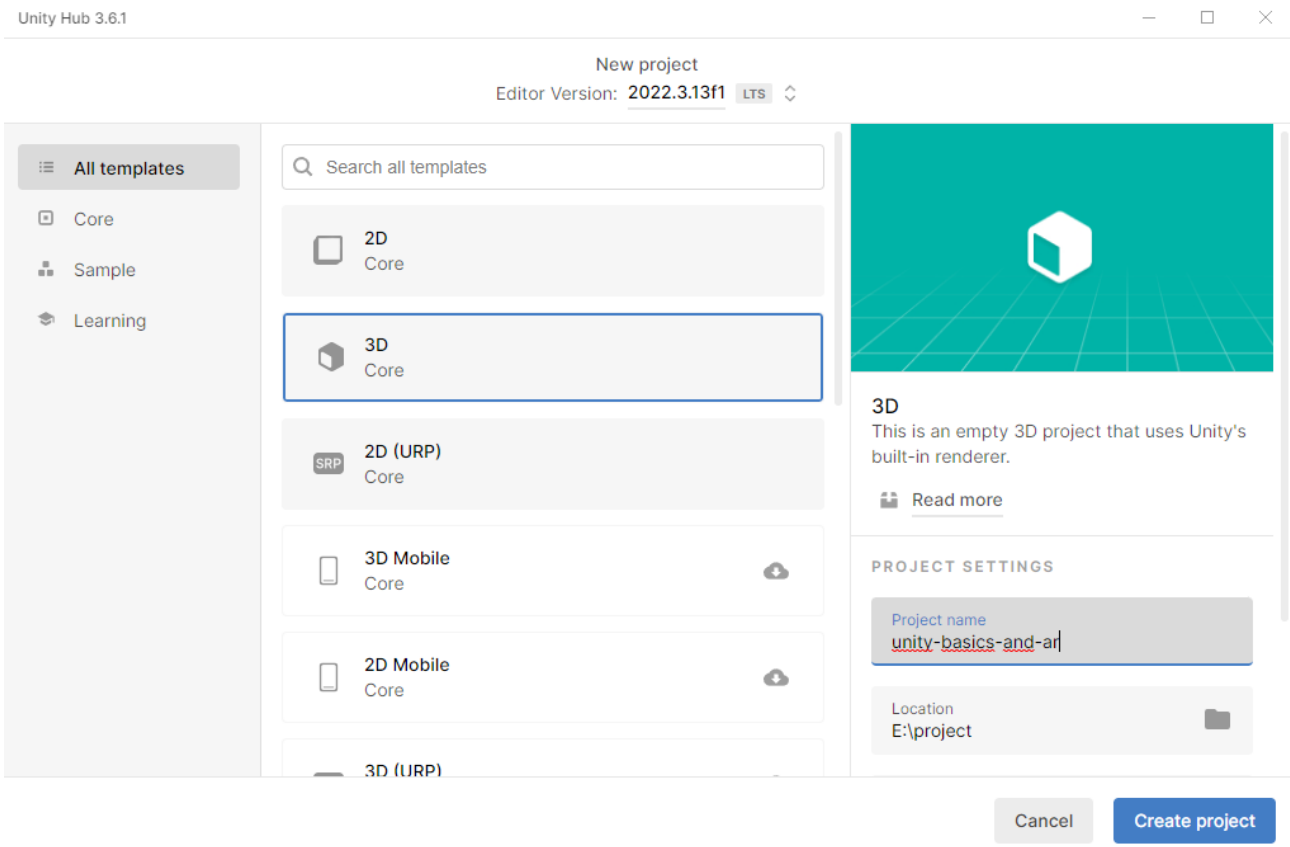
First Project

- Projects – New Project

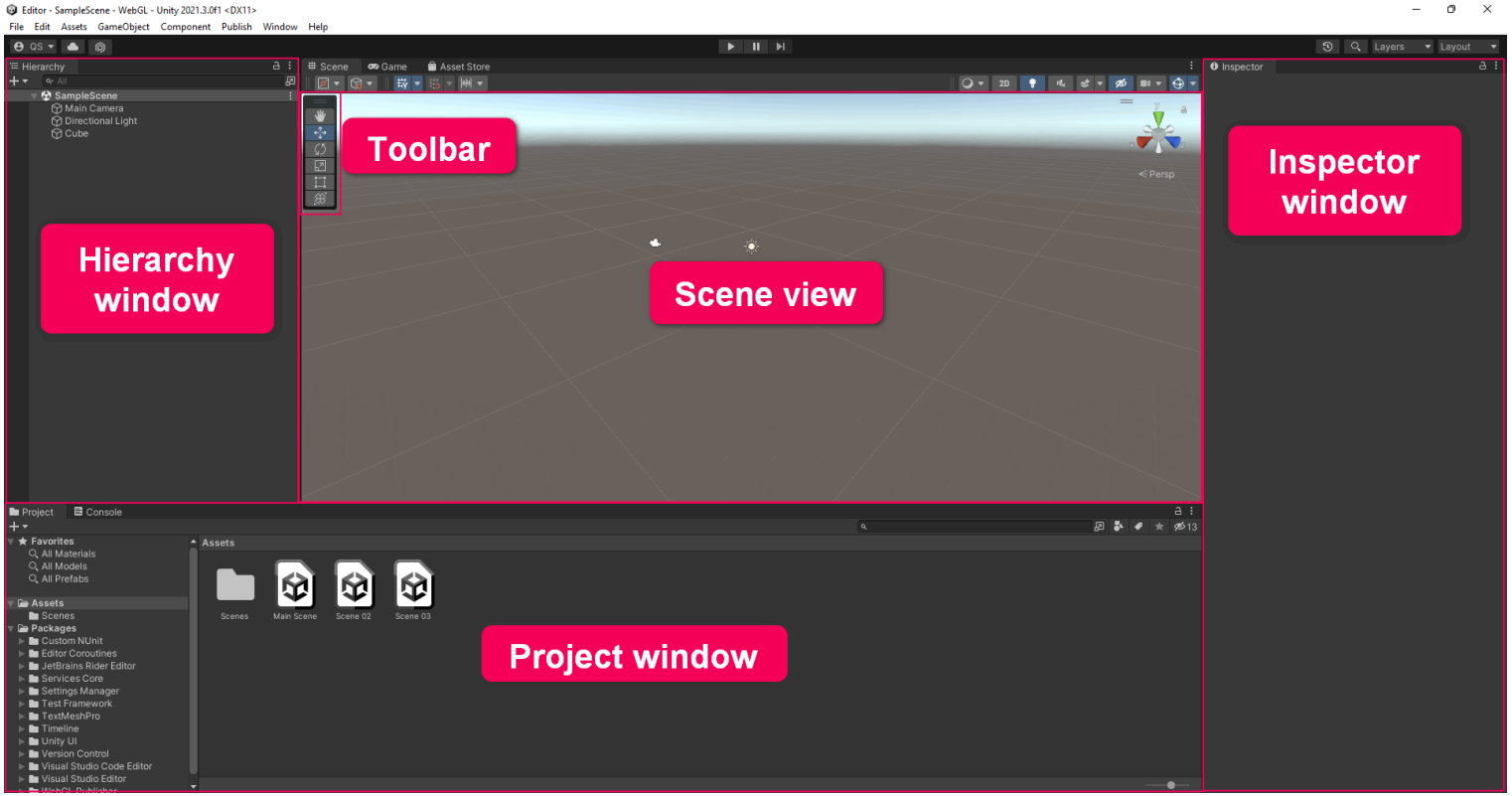


First Project

- Select editor version
- Select template
- Enter project name
- Set project path
- Click “Create project”
- Wait...



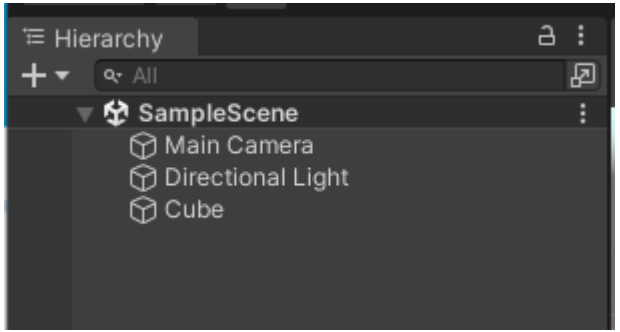
Unity Editor

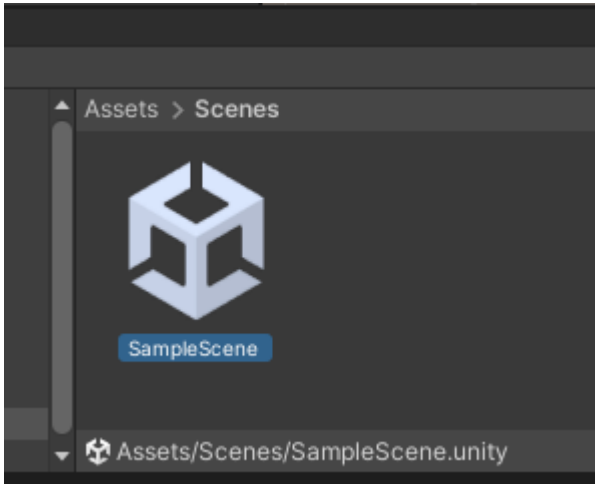


[10] – Expore the Unity Editor

Scene

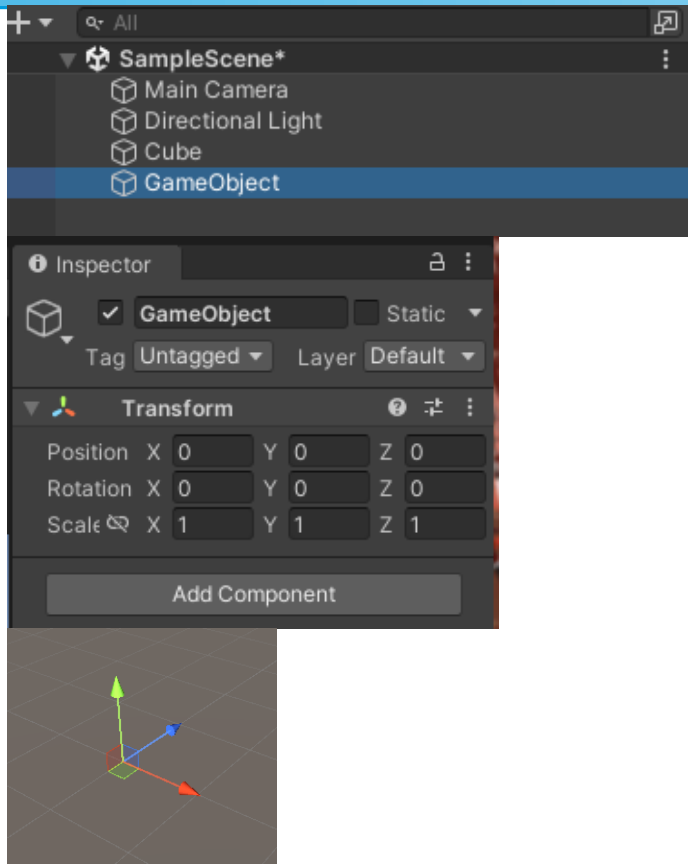
- Unity file that contains all **objects**
- Can visually **arrange** and **edit** the contents of a scene.
- Think of it like a **game level**
- You can **switch** between scenes.





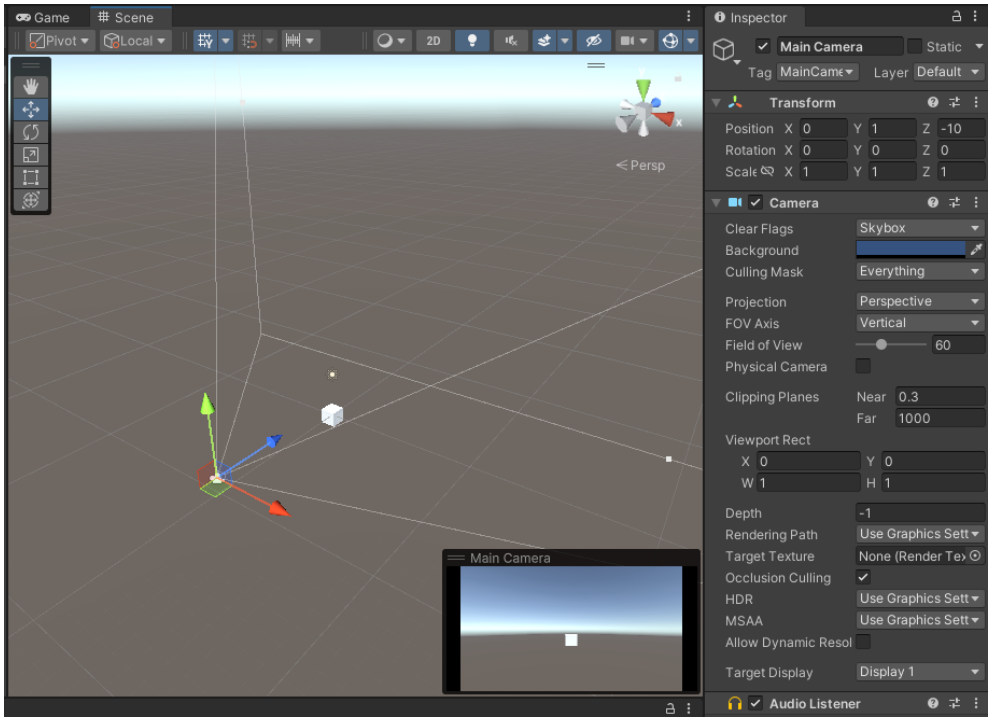
GameObject

- **Base** class for all entities in Unity Scenes
- Everything in the hierarchy is a **GameObject**
- All objects except canvas objects have a **Transform** component
- All canvas objects have a **RectTransform**
- Transform determines the **position**, **rotation** and **scale** of the object
- **Components** can be added



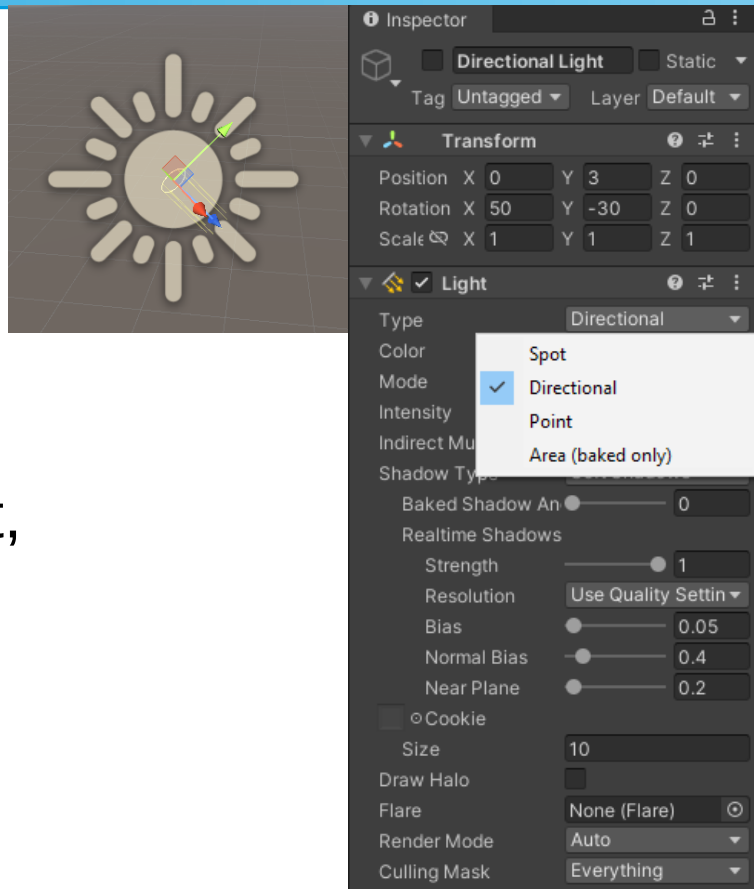
Camera

- Lens through the world
- **Audio Listener** attached by default
- It has **Transform** component which allows to manipulate object position, rotation and scale
- You can use multiple cameras



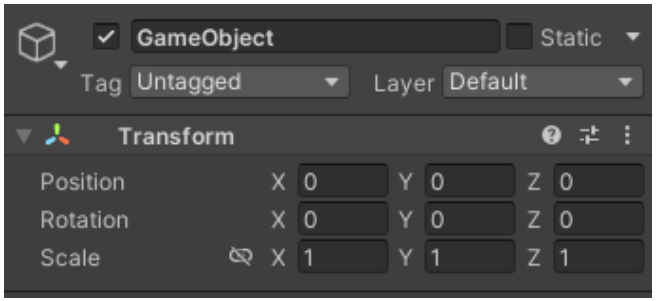
Light

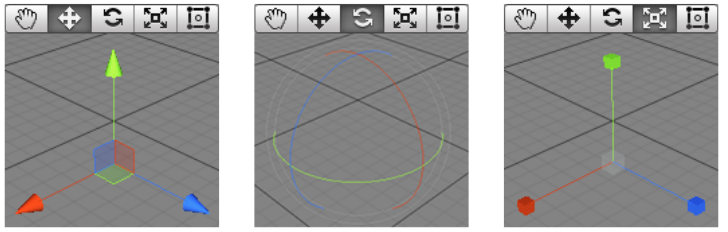
- Make objects **visible** to the cameras
- Make **shadows**
- Unity has **default lightning**
- **Directional Light** acts like the sun
- Ambient lightning, Point lightning, spotlight, Area(baked)



Transform

- Determines how the object will sit in space
- Child-Parent relationship
- **Translate**, **Rotate**, **Scale** operation





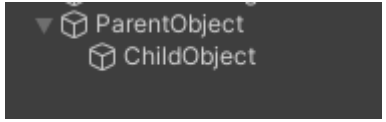
Translate (W) Rotate (E) Scale (R)

[11] - Unity Transform Tools

Parent-Child Relationship

- Transformations of objects are processed **relatively.**

  When the parent is **transformed**, the child is also **transformed**.
- You can edit parent-child relationships with **drag and drop** in the **hierarchy window**.
- Or with **script**.



Scripts

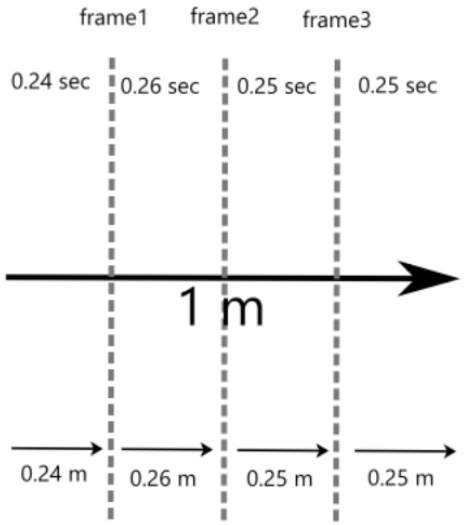
- Adds new **behaviour** to the object
- Inherits **MonoBehaviour**
- Exist as a **Component** of a GameObject

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

- **Start**: Called on the frame when a script is enabled
- **Update**: Called every frame, if the MonoBehaviour is enabled. Most commonly used function to **implement**

Time.deltaTime

- The **interval** in **seconds** from the last frame to the current one (Read Only)




```
using UnityEngine;
// Rotate around the z axis at a constant speed
public class ConstantRotation : MonoBehaviour
{
    public float degreesPerSecond = 2.0f;
    void Update()
    {
        transform.Rotate(0, 0, degreesPerSecond *
Time.deltaTime);
    }
}
```

[12] - Unity Time.deltaTime

Input

```
using UnityEngine;
using System.Collections;

public class ExampleClass :
MonoBehaviour
{
    void Update()
    {
        if (Input.GetKeyDown("space"))
        {
            Debug.Log("space key was
pressed");
        }
    }
}
```

[13] Unity Input.GetKeyDown()

- Input.GetTouch()
- Input.mousePosition()

```
using UnityEngine;
using System.Collections;

// Detects clicks from the mouse and prints a message
// depending on the click detected.

public class ExampleClass : MonoBehaviour
{
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
            Debug.Log("Pressed left-click.");

        if (Input.GetMouseButtonDown(1))
            Debug.Log("Pressed right-click.");

        if (Input.GetMouseButtonDown(2))
            Debug.Log("Pressed middle-click.");
    }
}
```

[14] Unity Input.GetMouseButtonDown()

Move Object

```
using UnityEngine;

public class ExampleClass : MonoBehaviour
{
    //movement speed in units per second
    private float movementSpeed = 5f;

    void Update()
    {
        //get the Input from Horizontal axis
        float horizontalInput = Input.GetAxis("Horizontal");
        //get the Input from Vertical axis
        float verticalInput = Input.GetAxis("Vertical");

        //update the position
        transform.position = transform.position + new Vector3(
            horizontalInput * movementSpeed * Time.deltaTime,
            verticalInput * movementSpeed * Time.deltaTime,
            0);

        //output to log the position change
        Debug.Log(transform.position);
    }
}
```

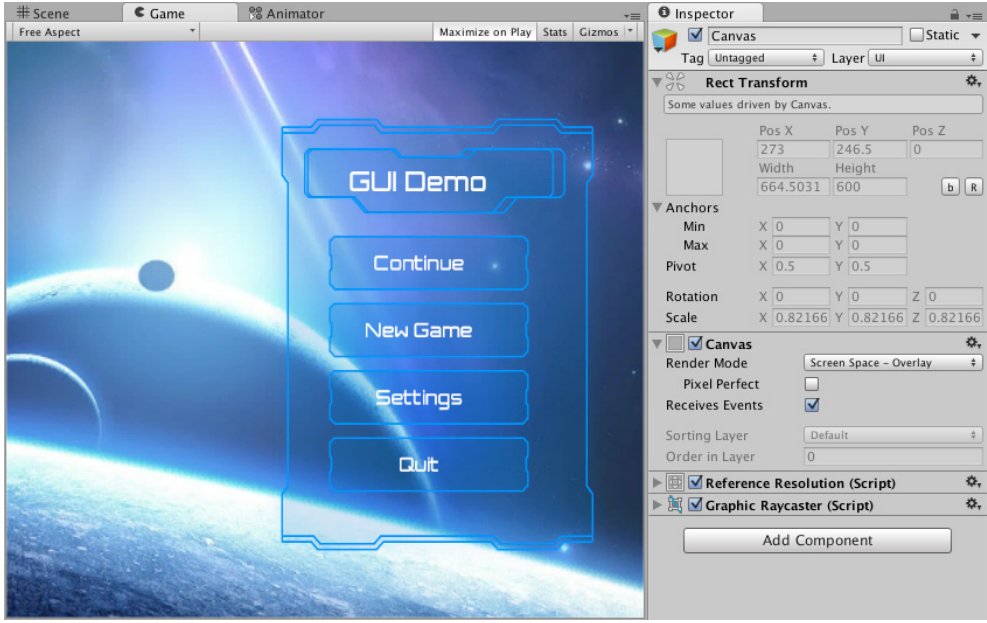
[15] – move with position

```
using UnityEngine;
using System.Collections;

// A very simplistic car driving on the x-z plane.

public class ExampleClass : MonoBehaviour
{
    public float speed = 10.0f;
    public float rotationSpeed = 100.0f;

    void Update()
    {
        // Get the horizontal and vertical axis.
        // By default they are mapped to the arrow keys.
        // The value is in the range -1 to 1
        float translation = Input.GetAxis("Vertical") * speed;
        float rotation = Input.GetAxis("Horizontal") * rotationSpeed;

        // Make it move 10 meters per second instead of 10 meters per
frame...
        translation *= Time.deltaTime;
        rotation *= Time.deltaTime;

        // Move translation along the object's z-axis
        transform.Translate(0, 0, translation);

        // Rotate around our y-axis
        transform.Rotate(0, rotation, 0);
    }
}
```

[16] – get axis

Canvas

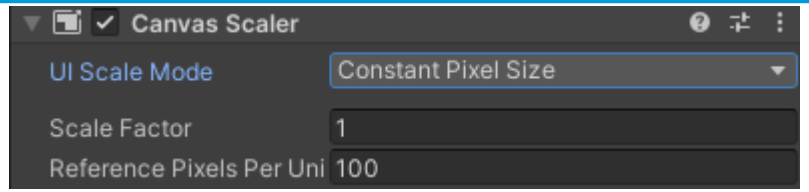
- The area that all UI elements should be inside
- All UI elements must be children
- The first child is drawn first
- 3 Render Mode
  - Screen Space Overlay
  - Screen Space Camera
  - World Space



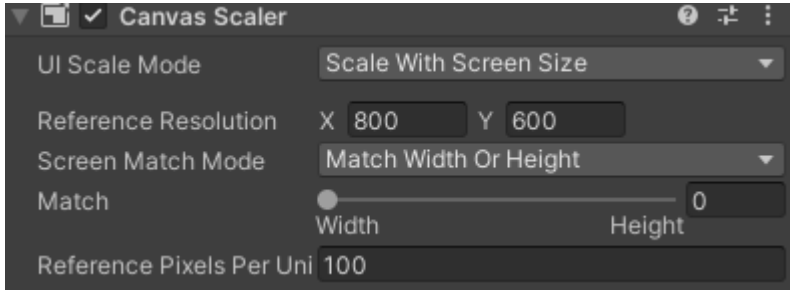
[17] – canvas Screen Space Overlay

Canvas Scaler

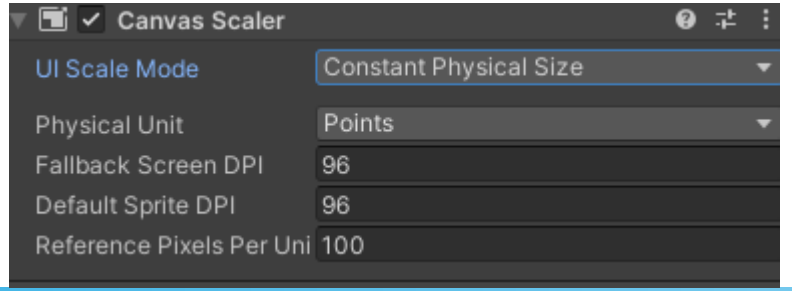
- Positions and sizes of UI elements are specified in pixels on the screen



- Positions and sizes can be specified according to the pixels of a specified reference resolution.

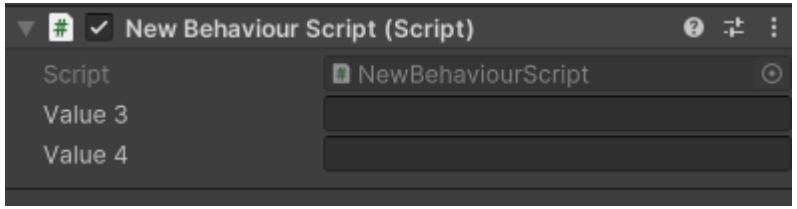


- Positions and sizes of UI elements are specified in physical units, such as millimeters, points, or picas.



Script Variable Visibility

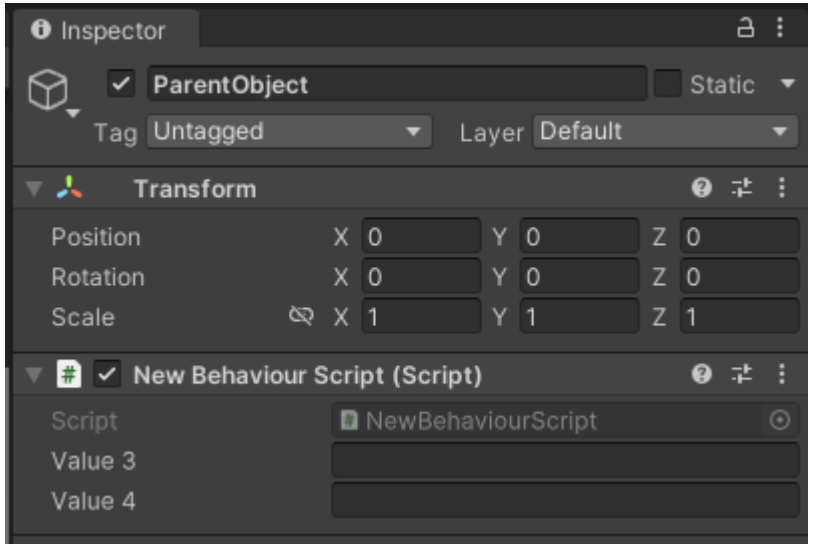
```
// visibility                // in class  // from another class  // Inspector
string value1;               //   yes             no                 no
private string value2;//   yes             no                 no
public string value3; //   yes             yes                yes

[SerializeField]
private string value4;//   yes             no                 yes

[HideInInspector]
public string value5;//    yes             yes                no
```



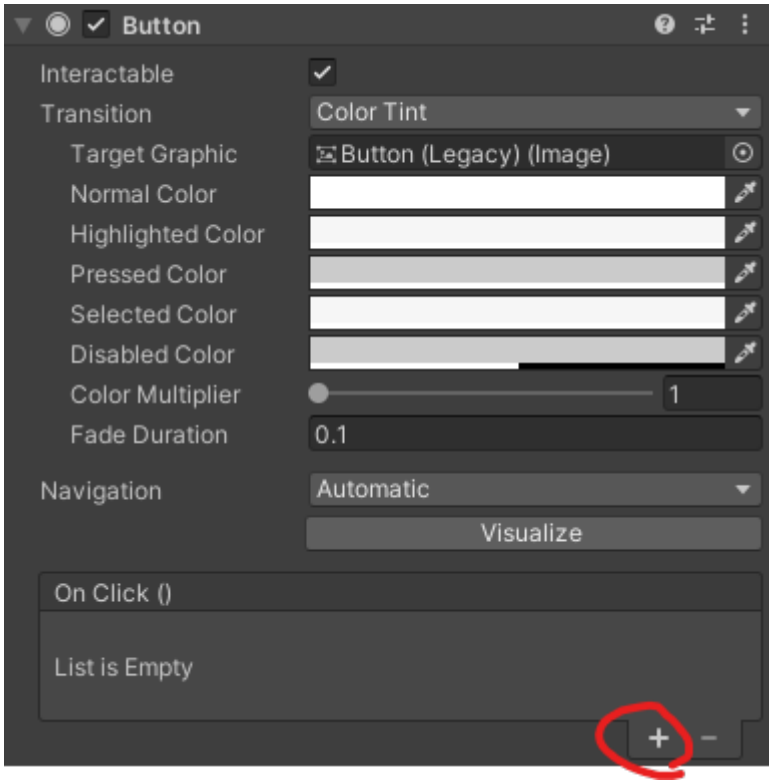
UI Button Event

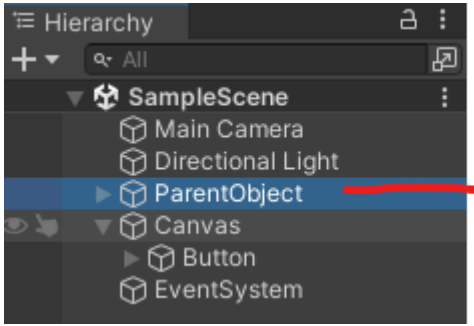
- You can use button events to control your system

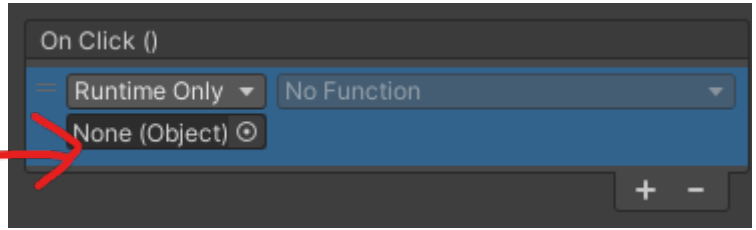
```
public void MyTestMethod()
{

}
```



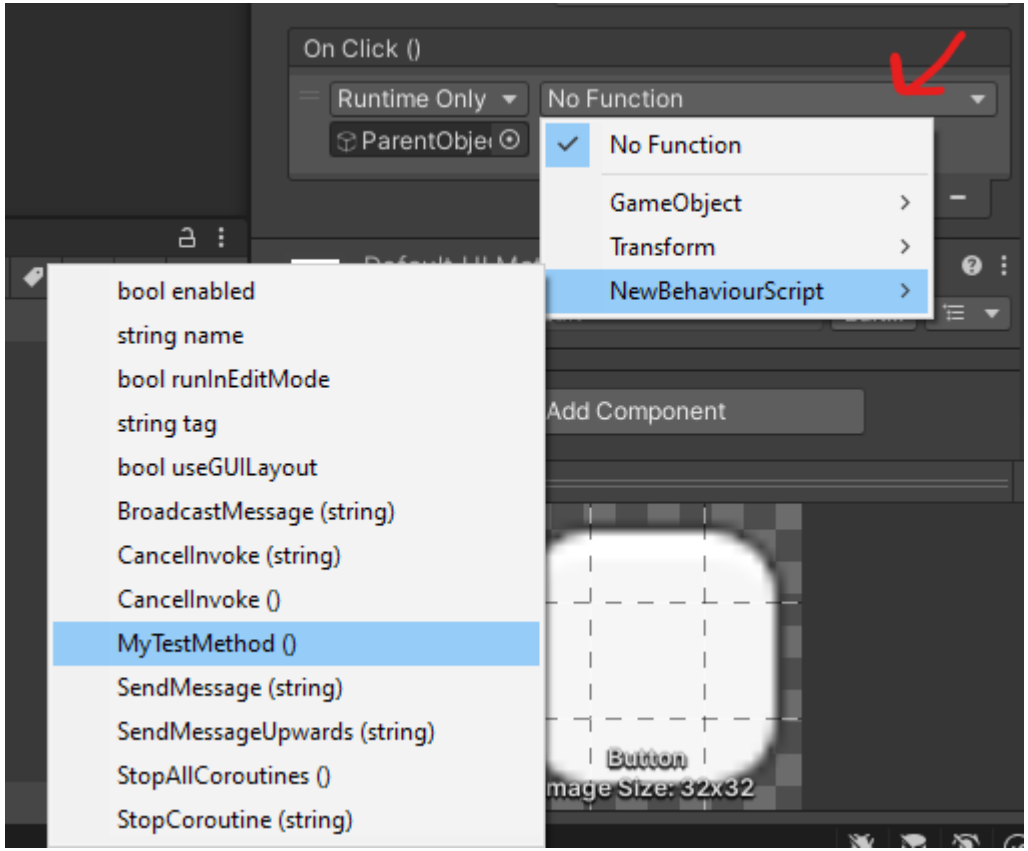
UI Button Event 2





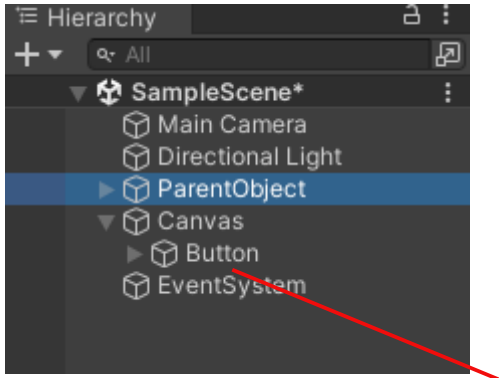


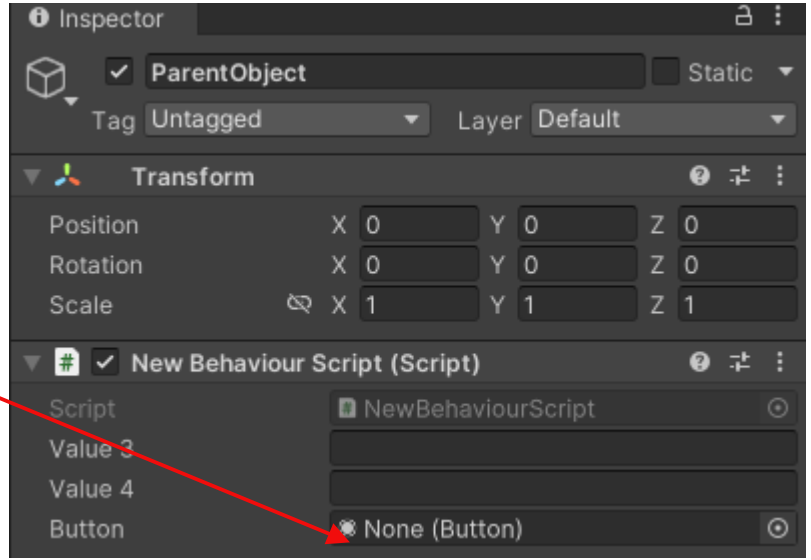
UI Button Event 3



UI Button Event 4

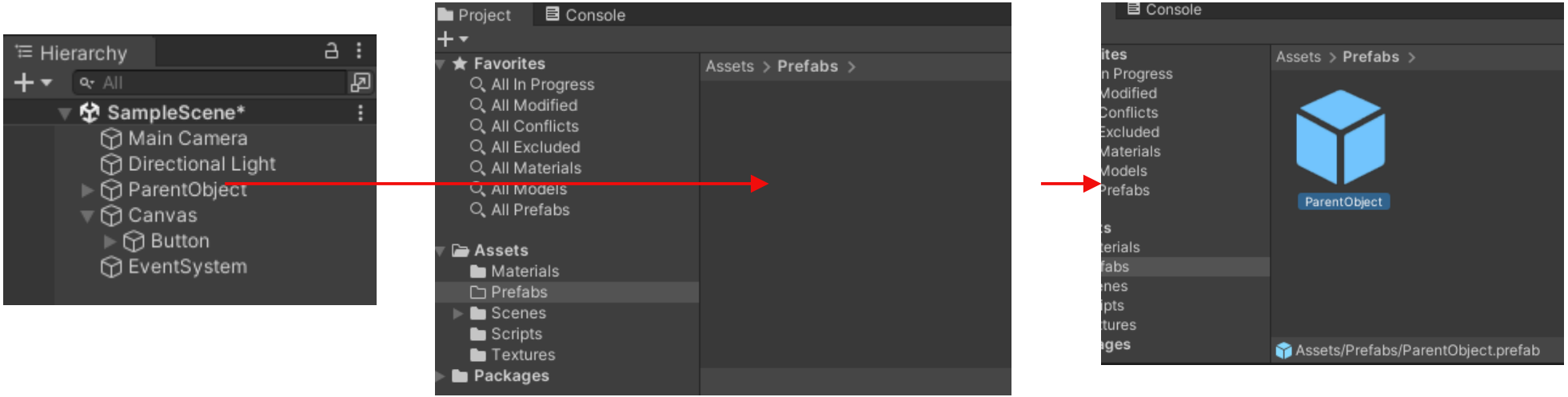
```
[SerializeField] private Button button;

void Start()
{
    button.onClick.AddListener(MyTestMethod);
}

void MyTestMethod()
{
    Debug.Log("Button pressed");
}
```





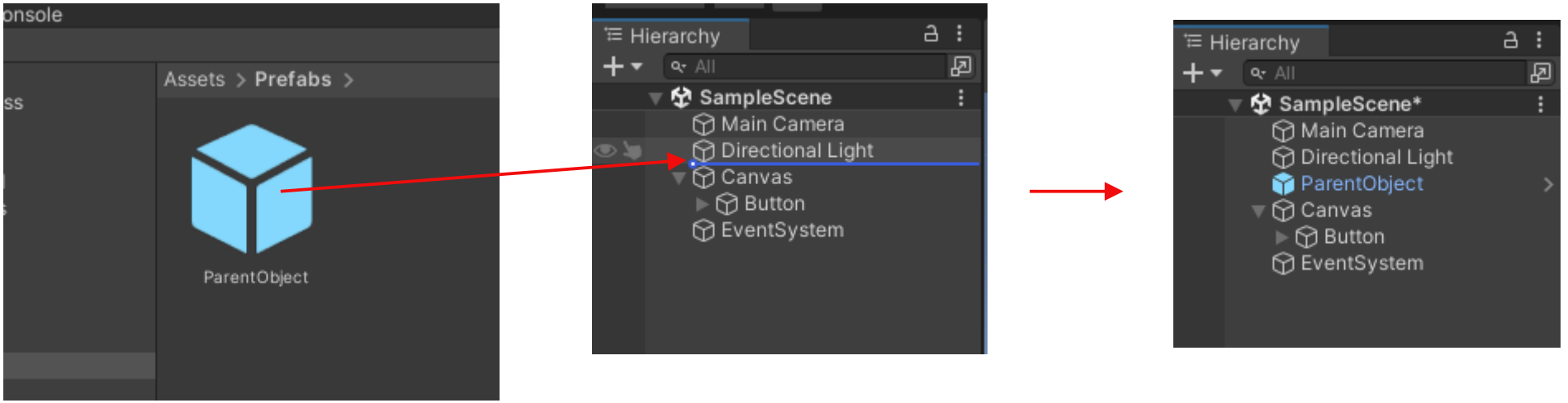
Prefab

- Store **GameObject** as a asset file in project folder to **reuse**



Drag game object from hierarchy to project window

Prefab Usage

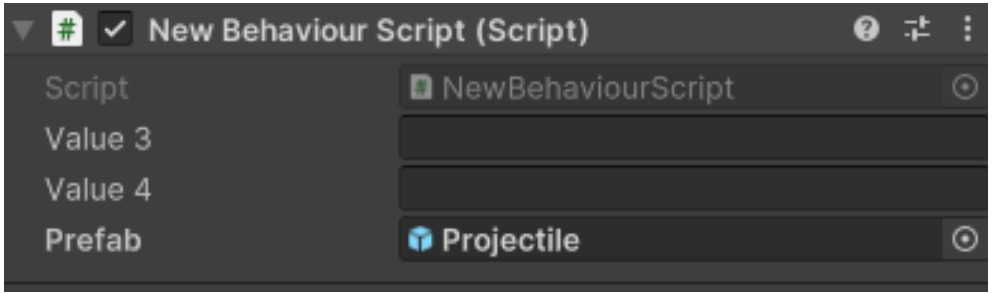


- Drag the prefab from project window to hierarchy

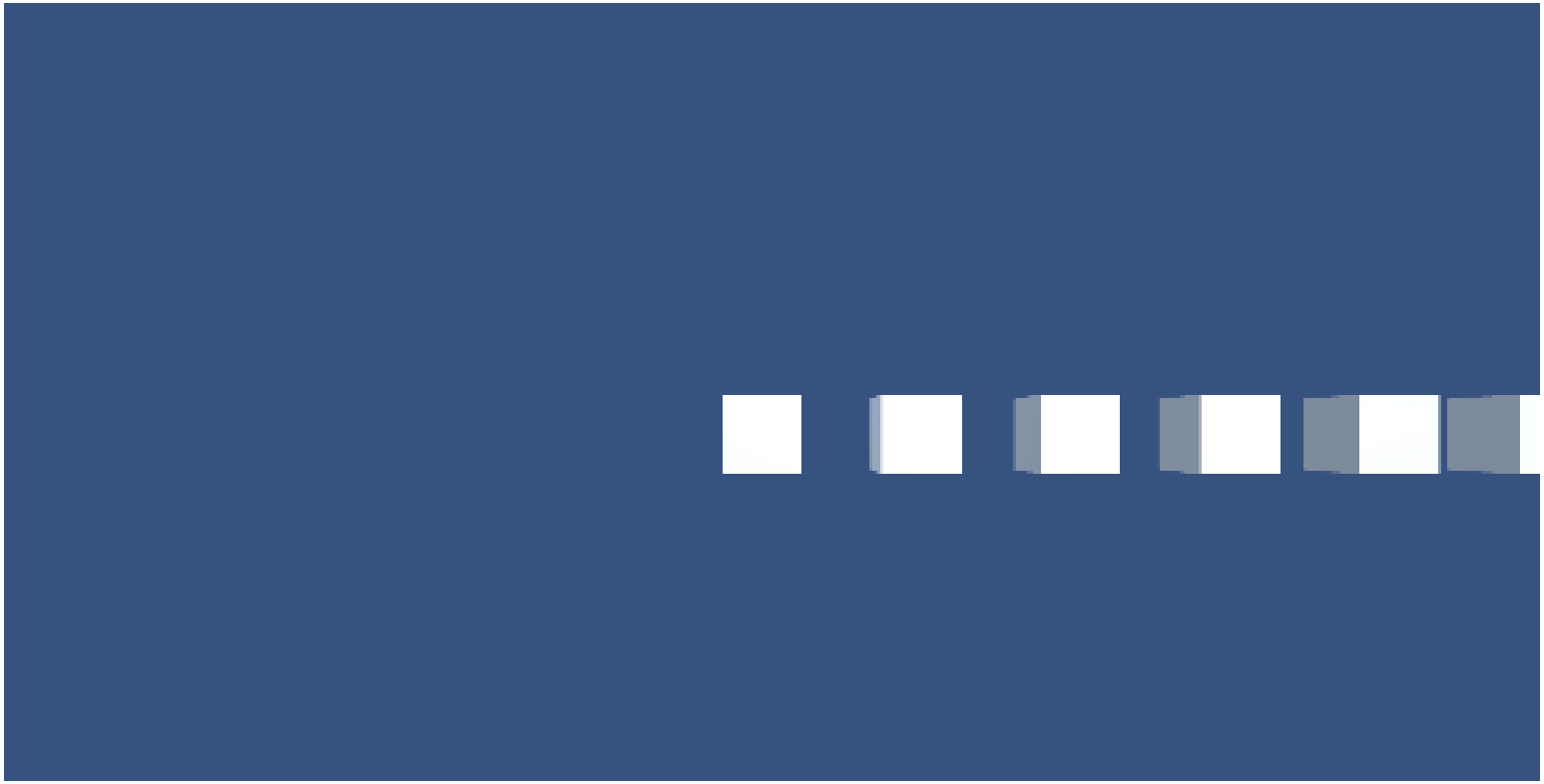
Object Instantiate

- Create a new **gameObject** with given **prefab**
- Add **GameObject** data member to your script
- Drag your **prefab** to script
- Call **Instantiate()** function

```
[SerializeField] private GameObject prefab;

void Start()
{
    for (var i = 0; i < 10; i++)
    {
        Instantiate(prefab, new Vector3(i * 2.0f, 0, 0), Quaternion.identity);
    }
}
```

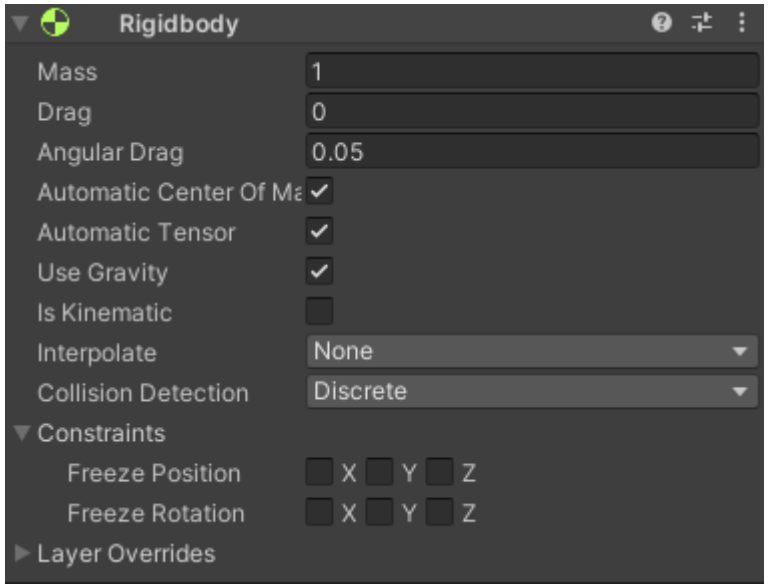


Object Instantiate 2



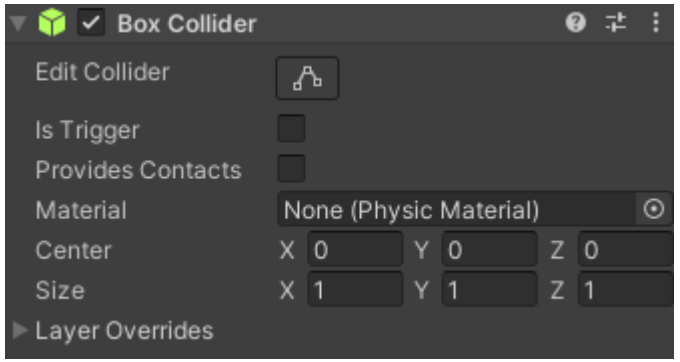
RigidBody Component

- Control of an object's position through **physics simulation** (ex gravity).
- You can apply **forces**
- Activate **collisions** for the object
- **Kinematic off** (**Collision** message, apply physic)
- **Kinematic on** (**Trigger** message, does **not** apply physics)



Collider

- Used for **collision**
- **Hitbox**
- Message collision events
- Works with **rigidbody**
- **Raycast** needs collider



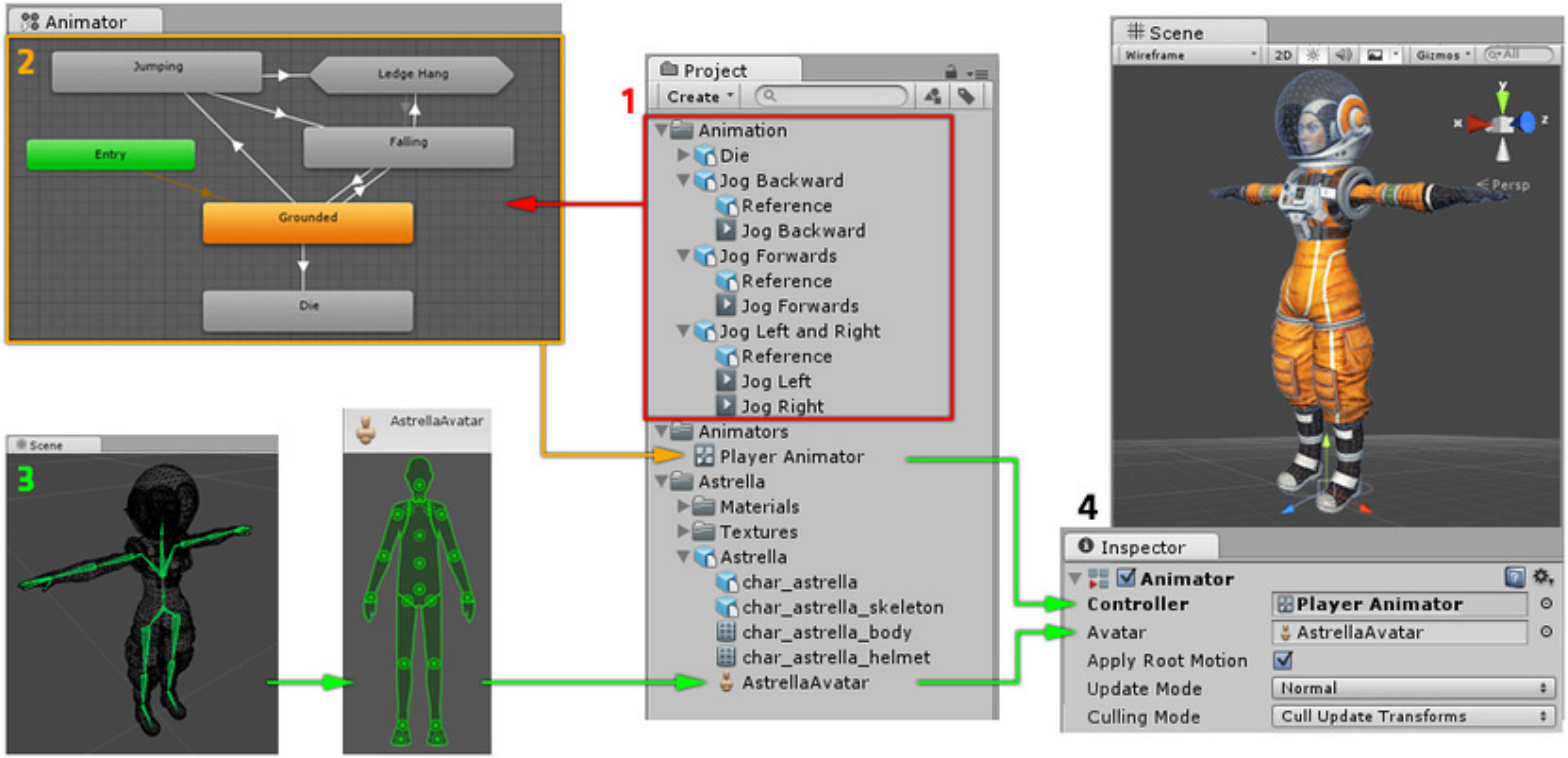
Raycast

- Casts a **ray**, from **point** origin, in direction **direction**, of length **maxDistance**, against all **colliders** in the Scene.

- Not detected if the Raycast origin is inside the **Collider**.

```
using UnityEngine;

public class ExampleClass : MonoBehaviour
{
    // See Order of Execution for Event Functions for information on FixedUpdate()
and Update() related to physics queries
    void FixedUpdate()
    {
        Vector3 fwd = transform.TransformDirection(Vector3.forward);

        if (Physics.Raycast(transform.position, fwd, 10))
            print("There is something in front of the object!");
    }
}
```

Animation



[18] – canvas Screen Space Overlay

AR Foundation

AR Foundation

- Create multi-platform **AR** apps
- Contains interfaces for AR features, but doesn't implement any features **itself**.

Unity officially supports the following provider plug-ins:

- Google ARCore XR Plug-in on Android
- Apple ARKit XR Plug-in on iOS
- OpenXR Plug-in on HoloLens 2
- Meta OpenXR Feature on Meta Quest

[19] – AR Foundation

Features

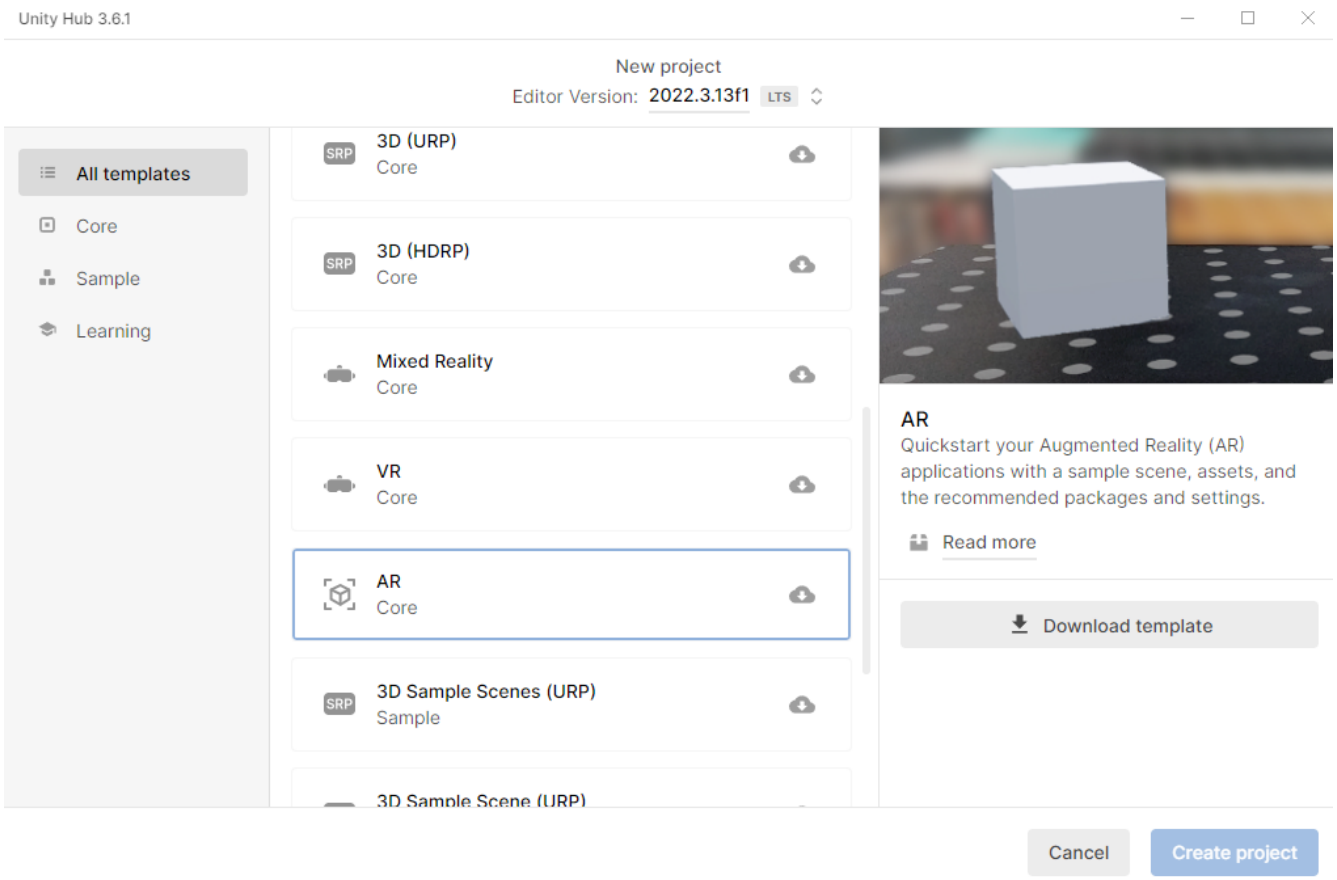
Unity's AR Foundation
Supported Features

Functionality	ARCore	ARKit	Magic Leap	HoloLens
Device tracking	✓	✓	✓	✓
Plane tracking	✓	✓	✓	
Point clouds	✓	✓		
Anchors	✓	✓	✓	✓
Light estimation	✓	✓		
Environment probes	✓	✓		
Face tracking	✓	✓		
Meshing			✓	✓
2D Image tracking	✓	✓		
Raycast	✓	✓	✓	
Pass-through video	✓	✓		
Session management	✓	✓	✓	✓

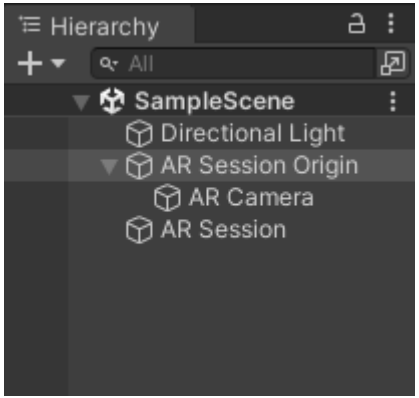
[20] – AR Foundation Features

Install

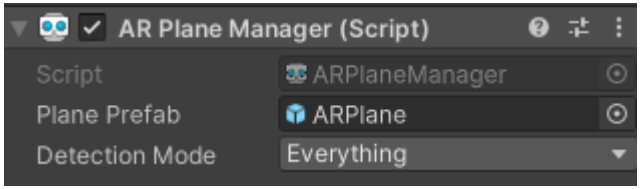
- Download template and create an AR Project

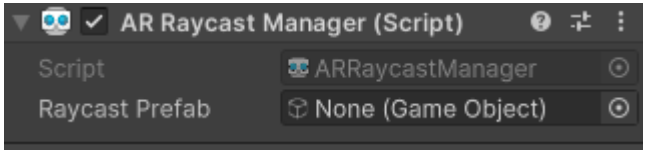


AR Classes


Hierarchy
All
SampleScene
Directional Light
AR Session Origin
AR Camera
AR Session


AR Session Origin (Script) (Depre
ARSessionOrigin has been deprecated. Use XROrigin instead.
Script
ARSessionOrigin
Origin Base GameObj
AR Session Origin
Camera Floor Offset (
None (Game Object)
Camera
AR Camera (Camera)
Tracking Origin Mode
Not Specified
Camera Y Offset
1.1176


AR Plane Manager (Script)
Script
ARPlaneManager
Plane Prefab
ARPlane
Detection Mode
Everything


AR Raycast Manager (Script)
Script
ARRaycastManager
Raycast Prefab
None (Game Object)

Placing Objects on Platforms with Raycast

- Add AR Session Origin
- Add AR Plane Manager
- Assign plane prefab to plane manager
- Add AR Raycast Manager
- Create a script for object placement

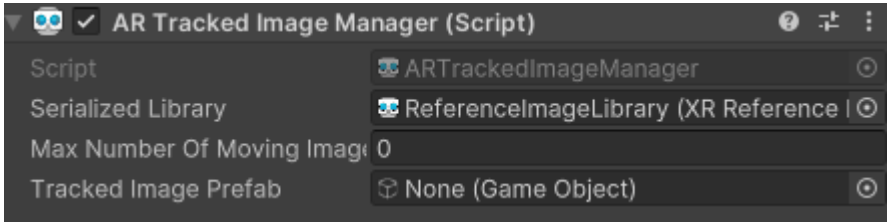
Placing Objects on Platforms with Raycast 2

```
[SerializeField] private GameObject prefab;
// Start is called before the first frame update

private ARRaycastManager _arRaycastManager;
private List<ARRaycastHit> hits = new List<ARRaycastHit>();
private bool _continuousTouch;

private void Awake()
{
    _arRaycastManager = GetComponent<ARRaycastManager>();
}
```

```
private void Update()
{
    #if UNITY_EDITOR
    // editor code
    if (!Input.GetMouseButtonDown(0)) return;
    var screenPos = Input.mousePosition;
    #else

    if (Input.touchCount != 1) return;
    if (Input.touches[0].phase != TouchPhase.Began) return;

    var screenPos = Input.touches[0].position;
    // device code
    #endif

    if (_arRaycastManager.Raycast(screenPos, hits, TrackableType.PlaneWithinPolygon))
    {
        foreach (var hit in hits)
        {
            Pose pose = hit.pose;
            GameObject obj = Instantiate(prefab, pose.position, pose.rotation);
        }
    }
}
```

Image Tracking

- Add AR Tracked Image Manager
- Assets > Create > XR > Reference Image Library
- Add image
- Assign your library and prefab



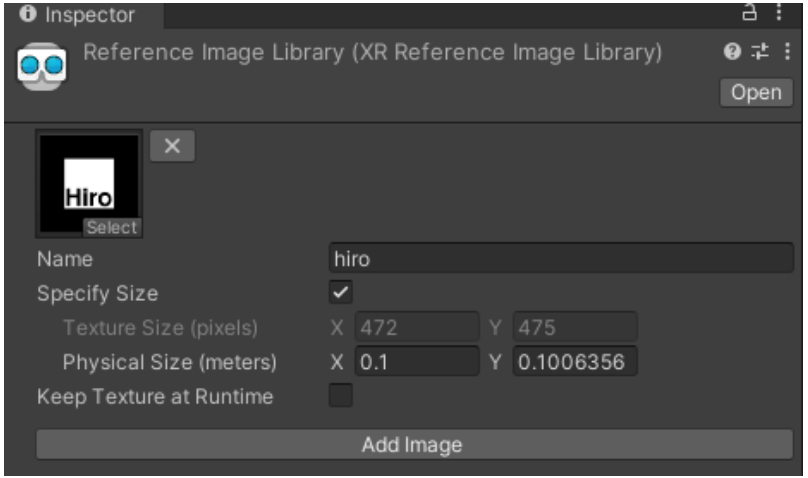


Image Tracking 2

```
[SerializeField] private GameObject prefab;

private ARTrackedImageManager _trackedImageManager;

private GameObject obj = null;

private void Awake()
{
    _trackedImageManager = GetComponent<ARTrackedImageManager>();
}

void OnEnable() => _trackedImageManager.trackedImagesChanged +=
OnChanged;

void OnDisable() => _trackedImageManager.trackedImagesChanged -=
OnChanged;
```

Manager will create a new image in the scene. You can use this image object to do what you want

```
void OnChanged(ARTrackedImagesChangedEventArgs eventArgs)
{
    foreach (var newImage in eventArgs.added)
    {
        // Handle added event
        if (obj != null) break;

        obj = Instantiate(prefab, newImage.transform.position,
newImage.transform.rotation, newImage.transform);
    }

    foreach (var updatedImage in eventArgs.updated)
    {
        // Handle updated event
    }

    foreach (var removedImage in eventArgs.removed)
    {
        // Handle removed event
        if (obj != null)
        {
            Destroy(obj);
            obj = null;
        }
    }
}
```

AR Project Example Links

- https://github.com/Unity-Technologies/arfoundation-samples/
- https://github.com/Unity-Technologies/arfoundation-demos
- https://learn.unity.com/project/create-a-marker-based-ar-app
- https://github.com/Unity-Technologies/XR-Interaction-Toolkit-Examples/tree/main
-

Useful Links

- https://stackoverflow.com/questions/62756029/does-a-vr-object-need-a-rigidbody-for-a-collider
- https://docs.unity3d.com/Manual/ExecutionOrder.html
-

Resources 1

- [1] - Unity Logo
  https://en.wikipedia.org/wiki/File:Unity_2021.svg
- [2] - AR Foundation Logo
  https://github.com/CodeChefVIT/resources/blob/master/AugmentedReality/ARFoundation.md
- [3] - Unity-chan
  https://unity-chan.com/images/imgKohaku.png
- [4] - Screenshot by me from a game named Genshin Impact
  https://genshin.hoyoverse.com/tr/ (official site)
- [5] - Simulation example image
  https://develop3d.com/features/unity-visualisation-vr-manufacturing-industrial-design-game-on-simulation/
- [6] - Data Visualize example image
  https://youtu.be/AESg_9F73uQ?si=nm0JYjXvgNp-ylVI
- [7] AR/VR Example image
  https://unity.com/how-to/best-practices-vr-and-mobile-ar-graphics
- [8] Film and Animation image from Sonder
  https://youtu.be/NHVqFt_UOkw?si=6roSYZ-2ecIdp8JE

Resources 2

- [9] - Education example image
  https://unity.com/solutions/edtech
- [10] - Explore the Unity Editor
  https://learn.unity.com/tutorial/explore-the-unity-editor-1#
- [11] - Unity transform
  https://docs.unity3d.com/2017.4/Documentation/Manual/Transforms.html
- [12] - Unity Time.deltaTime
  https://docs.unity3d.com/ScriptReference/Time-deltaTime.html
- [13] - Unity Input.GetKeyDown
  https://docs.unity3d.com/ScriptReference/Input.GetKeyDown.html
- [14] - Unity Input.GetMouseButtonDown
  https://docs.unity3d.com/ScriptReference/Input.GetMouseButtonDown.html
- [15] - move with position
  https://docs.unity3d.com/ScriptReference/Transform-position.html
- [16] - get axis
  https://docs.unity3d.com/ScriptReference/Input.GetAxis.html

Resources 2

- [17] - Canvas Screen Space Overlay
  https://docs.unity3d.com/Packages/com.unity.ugui@2.0/manual/UICanvas.html
- [18] – Animation
  https://docs.unity3d.com/2023.3/Documentation/Manual/class-Animator.html
- [19] – AR Foundation
  https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.1/manual/index.html
- [20] – AR Foundation feature
  https://unity.com/unity/features/arfoundation

Thank You